insertion point using a paste command. Also, the editor provides various commands (*e.g.*, "Paste =") to insert ~~new~~ a new node at the current insertion point.

Substitute the paragraph starting at page 13, line 32, with the following:

In one embodiment, the IP tree editor includes a tokenizer for receiving keyboard entered text, recognizing a token, and converting the token into a sequence of editor commands. The IP tree editor provides commands for selecting portions of an IP tree and for placing an insertion point in the IP tree. The IP tree editor allows various commands to be performed relative to the currently selected portion and the current insertion point. For example, the "Paste if" command replaces the currently selected portion of the IP tree by ~~an~~ a node pointing to the declaration node that defines the IP computational construct for conditional execution, which is generally denoted as "if" in current programming language syntax. The typing of an "if" token results in the corresponding "Paste if" command being executed, relative to the current selection.

Substitute the paragraph starting at page 16, line 1, with the following:

Alternatively, the crown can be selected by placing the mouse pointer over the operator and ~~the~~ then alt-clicking the left mouse button.

Substitute the paragraph starting at page 29, line 34, with the following:

In step 804, the routine displays the representation on the display device. The routine displays an insertion point indication at a position within the display representation that corresponds to the insertion point. In a preferred embodiment,

the routine preferably displays only a portion of long display representations, and allows the user to scroll within the display representation, causing the routine to display a different portion of the IP tree. The routine preferably generates in step 803 only as much of the display representation of the IP tree as is necessary to produce the displayed portion of the IP tree.

Substitute the paragraph starting at page 30, line 20, with the following:

In step 901, the routine allows the user to select an insertion point with reference to the display representation. In this step, the routine allows the programmer to select as the insertion point any position in the display representation that corresponds to either a node of the IP tree or a separation between nodes of the IP tree. The programmer may select the insertion point by pointing to the display representation using the pointing device. The programmer may also use the keyboard to select the insertion point by pressing a combination of cursor positioning keys that moves the cursor to the insertion point within the display representation. The programmer may also use an insertion point positioning command provided by the IP system. These commands move the insertion point to a given position in the IP tree, either absolute or relative to the current position of the insertion point.

Substitute the paragraph starting at page 30, line 32, with the following:

In step 903, the routine allows the programmer to select the type of node that should be inserted at the insertion on point. In a preferred embodiment, the programmer uses the keyboard to enter a token corresponding to a paste command

(*e.g.*, "if" for the "paste if" command). Alternately, the routine displays a list of paste commands for available node types and allows the user to select one. The routine preferably specifies the types of nodes which may be inserted as children of each node type. After the user has selected to paste a node type, the routine verifies that a node of the selected type may properly be inserted at the insertion point. This involves checking that a node of this type is a proper child of the node above the insertion point. If a node of the selected type may not properly be inserted at the insertion point, the routine preferably permits the programmer to choose another node type insertion point.

Substitute the paragraph starting at page 31, line 35, with the following:

The command well editor controls configuration of the user interface of the IP tree editor. The IP system provides a command well that contains each command provided by the IP system. In a preferred embodiment, each command has associated with it a unique identifier and function for implementing the command. A preferred command well is described in U.S. Patent No. 5,287,514, entitled "METHOD AND SYSTEM FOR CUSTOMIZING A USER INTERFACE IN A COMPUTER SYSTEM." The IP tree editor provides a user interface that includes a menu hierarchy, various toolbars, and various keymaps. The command well editor allows a programmer to customize the user interface. The command well editor allows a programmer to specify the menu hierarchy. A programmer can specify the names of menu items and the corresponding command to perform when the menu item is selected. The user interface preferably includes an arbitrary number of toolbars that can be individually and

optionally placed on any of the four borders of IP's main window (oriented horizontally or vertically) according to the user's preferences. The command well editor allows a programmer to specify which commands are to be included on the toolbars. Each command typically has a bitmap specifying a button for the toolbar. The user interface preferably includes an arbitrary number of keymaps that are chosen during the course of input based on the context of the current selection, *e.g.*, a left margin keymap for when the mouse is over the left margin area of the window and a program text keymap for when the mouse is over the program text area of the window. The command well editor allows a programmer to specify the mapping of a key (or mouse event) or sequence of keys to commands. For example, a double click in the program text area may map to a command to select a node, whereas a double click in the left margin may map to a command to display a subtree. Also, if a programmer maps the letters "if" (the "if" token) to the "Paste if" command in the program text keymap, then when the programmer types "if" within the program text area the "Paste if" command is executed which causes a node to be inserted at the current insertion point.

Substitute the paragraph starting at page 34, line 11, with the following:

Figures 11A-11D are sample diagrams showing the contents of a subtree and the display representation at various times during the insertion of two new nodes into the subtree. Figure 11A is a sample diagram showing the contents of the subtree and the display representation before any insertion has been performed. The subtree 1110 is defined by its root display node, a print node 1111, specifying a display operation. The print node is a child of another node, such as a grouping

node, not shown. An "A" variable reference node 1112 and a "IS THE ANSWER" string node 1113 are children of the print node. The insertion point left selection 1114, shown as an "|___" symbol, is located on the line above the print node. The extent of the left selection, as indicated by the shading of the nodes, includes nodes 1111, 1112, and 1113. The display representation 1120 contains a line of text 1121 corresponding to the subtree. An insertion point left selection 1122 is shown as a vertical bar cursor positioned at the beginning of the line and an ~~underscored~~ underscore indicating the extent of the selection.

Substitute the paragraph starting at page 35, line 16, with the following:

The programmer then selects the "B" variable reference node type to insert at the insertion point and inserts it. The "B" variable reference node may be selected and ~~insert~~ inserted by typing it directly from the keyboard or via a copy and paste operation. Figure 11D is a sample diagram showing of the IP tree and the display representation after the IP tree editor has inserted a "B" variable reference node as a child of the multiplication node. In the subtree, the IP tree editor has inserted a "B" variable reference node 1171 to replace the placeholder node. If "B" is entered via the keyboard, then IP editor will place the insertion point to just after "B" and leave the user in interpreted selection mode in anticipation that the expression may be extended by the user. This situation is illustrated in Figure 11D by the vertical cursor bar 1172 shown just after the "B" in node 1171. In the display, item 1180, the interpreted selection insertion point is shown as a vertical cursor bar, item 1174, with no explicit extent indication.

Substitute the paragraph starting at page 36, line 14, with the following:

Figure 12C is a diagram of the display list showing the expansion of display list entry 1224. The IP system selects display list entry 1224 to expand because it is the first unexpanded display list entry in the display list. Display list entry 1224 has been expanded into display list items 1228-1230. Display list entry 1228 is unexpanded, and its content indicator indicates the "A" variable reference node 1213. Display list entry 1229 is expanded, and its content indicator indicates a "*" string 1241. Display list entry ~~1224~~ 1230 is unexpanded, and its content indicator indicates the "B" variable reference node 1214.

Substitute the paragraph starting at page 38, line 6, with the following:

The IP system starts the reduction process by invoking the ProcessTE function passing the root tree element of a copy of the IP tree. The ProcessTE function loops invoking each Match function checking for a match and, when a match is found, invokes the corresponding Xform function. The Xform function typically replaces the pointer to the tree element it is passed with a pointer to a tree element that is the root of a reduced subtree and then recursively invokes the ProcessTE function to ~~processes~~ process the reduced subtree. For example, the Xform function for the _List IP computational construct recursively invokes the ProcessTE function each tree element pointed to by an operand tree component. When an Xform function has completed the reduction of the subtree that was passed to it, it returns the resultant (transformed) subtree which will be used by the calling function to replace one of the operands in its subtree, thereby completing one of the steps in transforming its subtree.